Rule Base Optimization for Agent-Based Simulations

Abstract

Agent-based simulation codes use rule bases to guide agents through a process. For example, "agents" could represent soldiers in a simulated military conflict, where the rule base defines when individual agents take various actions (e.g., advancing towards enemy lines or shooting at enemy soldiers). We develop a sequential algorithm for identifying rule bases which provide desirable outcomes, as well as giving an understanding of which rules are most important. By working in the space of contingency-table-like odds ratios instead of in the space where the rule base is directly defined (e.g., as a bit string with a finite alphabet), the fitness function behaves more smoothly and optimization is more easily pursued. The computational approach also nicely complements results from subjective visualization procedures.

KEY WORDS: Combat simulations, Emergent behavior, Odds ratio, Regression.

1 Introduction

Increased computing power in recent years has allowed for computer simulation of increasingly detailed processes. One class of such simulations involves individual "agents" which interact with each other and carry out various functions. In the motivating example for this paper, each agent is a soldier fighting in a simulated conflict. Other examples of agent-based simulations include such wide ranging applications as vehicles moving in a transportation network (Nagel, Rickert, and Barrett 1997), biological organisms evolving in the presence of threats (Jaffe et. al. 1997), individual investors behaving as part of a collective financial system (Deadman 1999), and products being assembled in an industrial manufacturing process (Fogerty and Bull 1995).

In such simulations, each agent consults a rule base that defines what actions take place at each time step. For the military example to follow, soldiers may move, adopt defensive postures, shoot at enemy soldiers, and so on. Of interest are properties of the system's emergent behavior, such as the probability that a particular army wins the battle.

To provide realism, agent-based codes are usually stochastic. In military simulations, simple examples of stochastic elements are line-of-sight probabilities (i.e., the chance that one soldier will see another who's in the vicinity) and kill probabilities (i.e., the probability that one soldier will kill another, given that he sees him and is shooting at him). More generally, stochastic elements in agent-based simulations can involve input file values defining initial conditions, agent travel time distributions, random system breakdowns and times to repair, and queueing phenomena. Owing to such stochastic behavior, statistical inference is essential.

Frequently, it is of interest to identify the best rule base governing agent behavior. Depending on the simulation involved, the term "best" could mean finding the rule base that gives an army its greatest chance to win a simulated battle, or that simulates traffic in best agreement with observed data, or that maximizes simulated widget production. Upon expressing rule bases in the form of trees or as fixed-length strings with finite alphabets, standard optimization techniques can be pursued, such as genetic algorithms (Steve: refn to Einstein application). As will be seen, however, rule base fitness behaves irregularly as a function of bit strings, which adversely affects the optimization process.

Moreover, it is not enough to obtain a "black box" solution providing little intuition regarding which rules matter and which ones don't. For purposes of using simulation results to support training, it is desired to identify a few basic rules for agents to follow. Because it is impractical to effectively teach an entire rule base covering all possible situations in which an agent could find himself, extracting important if-then relationships is essential. The approach described here achieves that goal.

The approach also differs from visualization in that results are scientifically reproducible. Although visualization allows the user to see relationships that might be difficult to discern otherwise, this strength can also be a serious weakness. In the extreme, visualization can be similar to psychology's Rorschach test, where people give highly personalized opinions as to what they "see" in ink blots.

In this paper, we describe a sequential method to extract useful rule bases for agent-based simulations, and apply it to a simplified problem. The approach is regression-based, can be implemented with standard statistical software, and yields results that nicely complement those from visualization or from optimization techniques such as genetic algorithms.

2 Combat Simulation

2.1 Background

To focus ideas, consider a simplified armed conflict between the good guys (more formally, the "blue team") and the bad guys ("the red team"). Given a set of initial conditions, simulations are performed to develop understanding of the conflict's dynamics. There are many such war gaming codes (Steve: need references).

In such codes, it is common to simulate individual time steps. During each time step, each agent evaluates his status, uses a rule base to determine what action(s) to take, and the time step is simulated, usually involving stochastic components – e.g., if one soldier shoots at another, the outcome is randomly generated. Of interest is the so-called emergent behavior of the system, which arises upon evolving the collective actions of the agents over the simulated time steps.

Emergent behavior is a function of the initial conditions for the conflict and of how the soldiers are equipped (e.g., their weapons, vehicles, sensors and properties thereof), as well as the rule base that guides their actions. In what follows we focus on the rule base, fixing the initial conditions and equipment. The goal is to find rule bases that maximize, to the extent possible, blue team performance.

At present, military analysts often carry out a vary-one-factor-at-a-time, seat-of-the-pants tweaking of rule bases, trying to identify important actions. In addition, available optimization procedures (e.g., genetic algorithms) are sometimes used to provide solutions. Though helpful, the output of such algorithms frequently has a black box quality, providing an optimum but no intuition about the importance of rules.

2.2 JIVES Town

We focus on a simulated urban conflict in a notional city called JIVES Town. The acronym JIVES (Joint Integrated Virtual Environment for Simulation) refers to a specific combat simulation code written in the Military Systems Group at Los Alamos (Upton 1999). This code is well suited to input/output for multiple rule bases in a manner compatible with statistical modeling. More importantly, it allows great flexibility in its bookkeeping, so that the user can track aspects of the simulation of personal interest, without being inundated with unwanted output. This feature is valuable with respect to calculating the odds ratios discussed later.

A terrorist-scale conflict is examined, where 8 blue agents square off against 4 red agents. At time zero, the blue agents start at locations specified depicted in Figure 1 with orders to advance towards city hall. They have 100 time steps to get there. Red agents remain in fixed locations and attempt to kill the blue agents when they come into range. More details about the simulation are given in the Appendix.

The status of each blue agent at each point in time is described by five binary variables:

(a) whether the agent senses one or more red agents in the immediate vicinity or not,

(b) whether the agent senses one or more blue agents in the immediate vicinity or not,

(c) whether the agent is under fire or not,

(d) the agent's current posture (offensive or defensive), and

(e) whether the agent has reached his goal, or not.

Regarding (d), the agent's posture, each agent may choose to be in offensive posture (which allows movement of a greater distance per time step) or defensive posture (which reduces vulnerability to enemy fire). Regarding (e), an agent is defined to be at his goal once he (Steve: fill in the blank, perhaps alluding to Figure 1).

The possible (binary) actions that each blue agent can take at each time step are

(a) if sense one or more red agents in the immediate vicinity, shoot at one of them, or not,

(b) change posture (e.g., from offensive to defensive), or not,

(c) if shooting and changing posture, decide in which chronological order these actions take place,

(d) move towards the goal at posture-dependent rate, or not, and

(e) look around the immediate vicinity (checking for presence of other agents to be sensed in the next time step) or not.

It is desired to find the rule base which gives the best average performance for blue team. That is, to determine under what circumstances blue agents should look, move, adopt defensive postures, and so on. The rule base for the red team is fixed throughout (see appendix), avoiding for now problematic game-theoretic issues where the red team modifies its own rule base to counter changes in the blue team's tactics.

The human analyst specified, a priori, a rule base that seemed to him to be optimal based on experience with other simulation codes. That rule base is displayed in Table 1 and specifies for each blue agent to always look, always move, to shoot if sense red and in defensive posture, to adopt defensive posture and shoot if sense red and in offensive posture, and to adopt offensive posture if don't sense red and are in defensive posture. Our goal is to find better rule bases, if possible, and to understand the importance of taking various actions under various conditions.

2.3 Modeling the Data

In the approach discussed below, we use regression modeling to aid in discovering good rule bases. Because there is no first-principles theory for emergent behavior, the regression model is an empirical one.

We take as the response variable the average number of blue agents who reach the goal and are alive at the end of the battle. In a sense, this response is equivalent to the probability of a single blue agent's success (just divide by 8). An initial tendency is to model the response as a function of covariates. As is apparent from Table 1, a rule base is completely defined by a string of 160 bits, detailing which of the 5 actions are taken under each of the 32 possible conditions an agent might face in a given time step. Models based on the related 160 bits could be considered. Upon allowing model parameters for combinations of bits to explore effects of actions taken 2 and 3 at a time, such models contain literally thousands of terms. If it were practical to obtain results from millions of rule bases, models of this form could be pursued, together with the dimension reduction techniques required to distill the results into something interpretable.

Such an approach is tractable in the JIVES Town setting when there is access to high performance computing, but not in more realistic problems where the number of status/action variables is much larger than the 10 at present, such variables may be polytomous (vs. binary) or continuous, and different subsets of agents can have their own respective rule bases. Thus, in more realistic simulations there are far more rule bases than the 10^{35} in the current problem (meaning that the space to be searched is much larger), and there is the double whammy that the additional real world complexity means that a single simulated battle requires far more CPU time than for a single JIVES Town run. Consequently, making millions upon millions of runs is generally impractical.

With an eye towards making progress in more complex settings, we choose to model the fitness function in terms of other output responses instead of modeling it in terms of rule base descriptors such as bit values in a bit string. The response variables we choose are more meaningful and their space has much smaller dimension than for combinations of bits. In addition, the fitness function behaves more smoothly as related to those responses than with respect to bits in a bit string, where altering a single bit can greatly affect fitness.

Regressor variables to follow are formed by using log odds ratios of a rule base's empirical behavior. For the i^{th} (of 5) action variables in Table 1, define its "main effect" as the odds ratio

$$\alpha_i \;=\; \frac{(\#\text{ times take action } i\ +0.5)/(\#\text{ times could have taken action } i\ +1)}{(\#\text{ times didn't take action } i+0.5)\ /(\#\text{ times could have taken action } i\ +1)}\ .$$

If all 8 blue agents were to remain alive for all 100 time steps in all r replicated runs for the rule base, then the number of times that action i could have been taken would be $8 \times 100 \times r = 800r$. Main effects for the 5 status variables are defined similarly, counting the number of times that each condition occurred and didn't occur. The factors 0.5 and 1 in the odds ratio expression for α_i are included to avoid numerical problems with zero counts.

Regressor variables are next constructed for the "two-factor interactions" between variables. For action variable i and status variable j, for example, this interaction is summarized by the odds ratio

$$\beta_{ij} = \frac{(\#\text{ times take action } i \text{ when in status } j+0.5)/(\#\text{ times in status } j+1)}{(\#\text{ times don't take action } i \text{ when in status } j+0.5)/(\#\text{ times in status } j+1)}$$

$$\times\frac{(\#\text{ times don't take action } i \text{ when not in status } j+0.5)/(\#\text{ times not in status } j+1)}{(\#\text{ times take action } i \text{ when not in status } j+0.5)/(\#\text{ times not in status } j+1)}\ .$$

The term β_{ij} is related to conditional independence structures in contingency tables (see, e.g., Bishop, Fienberg, and Holland 1975, p. 13). That is, consider the 2×2 table having row labels "take action i" and "not take action i" and column labels "in status j" and "not in status j". Cell counts for the table denote the total number of times during the simulation of the rule base that blue agents were in the status indicated and acted as described. From these counts β_{ij} is computed. Were β_{ij} to equal zero, for example, there would no relative preference, in an overall sense, for agents to take action i more/less often when in status j than when not in status j.

The set of odds ratios $\{\alpha_i\}, \{\beta_{ij}\}$, and their counterparts for higher-order interactions summarize, in an aggregate sense, the behavior of agents who adhere to the rule base. More thoroughly, odds ratios could be extended to summarize sequences of actions covering 2 or 3 time steps, in an attempt to understand complex behavior.

[Note. Odds ratios from rule bases are not interpretable in exactly the same way as for contingency tables. The cell counts in a table of rule firings don't evolve from an i.i.d. sampling process. Also, some care in interpretation of these odds ratios is occasionally required owing to Simpson's paradox issues (Samuels 1993) for collapsed tables. Nonetheless, we have found odds ratios to be valuable in model fitting.]

Our approach is to regress the average response for each rule base on the log odds ratios. Questions such as "do rule bases whose agents move frequently perform better than rule bases whose agents don't?" can then be directly addressed by examining parameter estimates from the fit. For example, suppose that the estimated coefficient of the log odds ratio β_{ij} for the action variable "move" and status variable "under fire" were negative and deemed significant. This would indicate that rule bases performed better when their agents moved less often when under fire than when not under fire. Although such information doesn't completely determine under which conditions agents should move, it helps define a good set of rule bases for future consideration.

2.4 Searching Rule Base Space

Search techniques should be tailored for the problems to which they are applied. Important aspects of the surface to be optimized are the following.

(a) Expressing the blue team rule base in the form of Table 1, there are 32 possible status conditions at each time step and 8 to 32 possible actions for each condition (recall the constraints that an agent must sense an enemy in order to shoot, and must be shooting and changing posture in order to choose the chronological order in which these two actions take place). Thus, the space of allowable rule bases contains roughly 10^{35} members, precluding anything approaching an exhaustive simulation of all allowable rule bases.

(b) The code is stochastic, so that multiple runs of the code are needed to estimate the average performance of a rule base.

(c) The vast majority of allowable rule bases give terrible performance, with few if any blue agents reaching the goal.

(d) The surface is highly irregular when expressed as a function of bit string representation; e.g., changing a single action for a single status in a rule base can greatly alter performance.

The bottom line is that it is impossible to guarantee that a global optimum will be found through *any* optimization procedure. A realistic goal is to understand the if-then relationships important to good performance.

Despite the above complications, the problem is well suited to genetic algorithms, which are adept at searching spaces of bit strings. This approach has been pursued with the Einstein code (Steve: refn), but the black box nature of the solution often doesn't help in understanding which rules are important and which aren't. Moreover, the nature of mutations in genetic algorithms is sometimes such that the algorithm makes only local moves from generation to generation, which inhibits widespread search of the large space of rule bases.

In order to generate data for regression modeling, it is necessary to simulate output from several rule bases. An obvious question is how to choose those rule bases. In what follows, we take a sequential approach, common to statistical applications of experimental design in other settings. That is, we start with an initial set of rule bases, analyze the results, choose another set of rule bases for further investigation, and so on.

For the first iteration of this sequential process, we choose a space-filling design. Many approaches to filling bit string space exist. In what follows, we construct random rule bases by letting each bit be one or zero independently with probability 1/2 each. (One might also contemplate forcing a kind of balance in the design by ensuring that certain components of the bit strings take on each of their values equal numbers of times.) In our first iteration, we selected 640 rule bases, and ran each of them ten times. This required [less than 24 hours] of computing time. Since the code is stochastic, replications of rule bases are necessary, but in this problem, many randomly generated rule bases achieve a zero response variable with probability essentially one, while good rule bases will perform clearly better even after only ten runs. At this early stage, few runs per rule base will be sufficient to help us tell the difference between very bad and somewhat promising rule bases; later we will require more runs in order to detect more subtle differences. This is especially true since variability increases with average response so that the rule bases of greatest interest also have the largest variability.

This design falls far short of actually filling the space of $2^160 \sim 10^{35}$ possible rule bases, but it is nevertheless capable of finding some promising paths. An artifact of space filling designs in this context is that the vast majority of rule bases chosen randomly give poor blue team performance: in our design, 557 out of 640 (87%) failed to get any blue agents to their goal, and only 24 rule bases averaged at least one agent per run. This is not necessarily bad, in that it's as often helpful to learn what rules an agent should *not* follow. When combined with data from good rule bases run in later iterations of the sequential design, data from poor rule bases aid in parameter estimation. Also, one of the 640 runs yielded an average of 3.6 blue agents at the goal, which is very close to the rule base constructed by the expert analyst.

In early iterations of the design, parameter estimation is highly variable. Because most of the rule bases are poor ones, the model fitting process is similar to fitting outliers. Owing to multicollinearity in the odds ratios, certain coefficients can have the wrong sign at this stage.

A goal of the analysis of the first design iteration, is to identify main effects (and hopefully some two factor iteractions) whose signs we are confident of (e.g. it is always good to shoot). These effects can lead us to fix certain components of second and later designs so that our searches are concentrated locally in promising regions. In our example, the coefficient for "Shoot" is highly significantly positive, so in our second iteration, we run 64 rule bases which are randomly generated except that their agents shoot at every opportunity. Less fortuitously, the first run has a highly significant negative coefficient for "Look" which causes us to run a number of hopeless rule bases in which agents never look. (This error seems to occur because of the confounding of shooting and looking: it is not possible to shoot unless you have just looked. Rules which shoot often also look often, and some of the credit for successful rules which is due to looking is given to shooting).

In an ideal world, the sequential design and analysis could be fully automated. Unfortunately, there aren't any good expert systems around for doing regression on the fly with decent diagnostics (i.e., looking for data transformations, outliers, and so on). Thus, we proceed with a manual implementation, postponing automation until a later date.

3 Data Analysis

By its nature, emergent behavior is complex. Although a rule base may be conceptually simple to specify, the evolving interactions of agents can produce emergent behavior that are sometimes difficult to understand. Thus, the model fitting to be done is by necessity purely empirical, intended to identify important if-then rules.

model fitting done here, using ordinary least squares, gives a rough approximation; heteroscedastic data (replicate runs indicate that variability increases as a function of the average response); logit transform didn't do much for us (in region of interest, the response variable is relatively stable, say from 3.5 to 4.5 of 8);

there's no reason to believe that a simplified regression model will fit the data, of course, but the prospects of using canned software is attractive, and the goal is just to search the space; may have problems with multimodal surfaces; can use R^2 value as rough indicator of model fit, indicating that we're getting reasonable predictability; can do a lack-of-fit versus pure error here, which gives modest success; good news is that things are a lot smoother than in bit string space;

as is apparent, any hypothesis testing to determine which model terms are "significant" isn't methodologically pure; recall, however, that the goals are to guide the iteration to find good rule bases and to obtain a rough idea of the relative effects;

3.1 From modeling to the next design

goal is not only to identify an optimal rule base (to the extent this is possible from a limited number of runs) but also to provide guidelines for training; a problem with standard optimization approaches is that they often don't explicitly convey which rules are important and, in most training applications, will only be able to train on a few simple rules and won't be able to impose an entire rule base; to be sure, there are cases like transportation simulation, might be able to simple lift a rule base into a code;

need to build up data base to overcome aliasing; over time, have a data mining situation, where have run numerous rule bases and obtained output; remember that the goal of the above is to find something different than existing approaches for analysts to study; the highly multicollinear nature of the odds ratios makes for difficult interpretation at times;

good (optimal?? see if it matches the best one) solution:

(a) always look,

(b) move most of the time (if not at goal and under fire, then don't move),

(c) if in offensive posture, change to defensive posture,

(d) if sense red, then shoot,

rationale: offensive posture is a loser, since are moving fast enough to get to goal otherwise and it gives up an advantage to red in one-on-one exchanges because of the greater chance of being hit when shot at; not moving when under fire allows other blue agents to catch up and to allow any dead blue agents nearby to confuse the red agents, thereby increasing the chances of success;

coefficient estimates to quantify the relative merits of the above rules; even if had the coefficients for the optimal rule base, wouldn't be able to do a one-to-one transformation to the underlying rule base; make a comment about the fitted coefficients for the data here;

note that there are no model terms with "at goal" involved; status is too highly correlated with the response to be of much use in an analysis; also, under the conditions here, virtually the only way for a blue agent to reach the goal is for all the red agents to be dead; thus, it doesn't matter what a blue agent does or doesn't do once he gets to the goal;

more detailed modeling, such as looking at *which* agents succeeded and which ones didn't, would reveal more info but at a severe cost in analysis time;

coefficient estimates to quantify the relative merits of the above rules; even if had the coefficients for the optimal rule base, wouldn't be able to do a one-to-one transformation to the underlying rule base; make a comment about the fitted coefficients for the data here;

interaction terms are defined by odds ratios in higher-order tables, not by multiplying the x values from lower-order terms, as in ordinary response surface models; can have lots of zeros when conditioning on combinations of events;

note that this solution is better (by 0.8 men to goal, on the average) than the supposed expert rule base, a rule base which might have advantages under more realistic conditions such as shooting ability degraded when on the move, or when there's a premium on getting to the goal quickly); in any case, the modeling approach yielded something different than what the expert initially started with;

shouldn't confuse frequency of rule firing with importance;

the under fire and don't move tends to improve clustering of blue agents, which here is a good thing; of course, if red agents had hand grenades, clustering could be a bad thing; this reflects the scenario-specific nature of an optimum;

would be nice to compare efficiency with that of a genetic algorithm, seeing how many code runs it needs to get a decent optimum;

4 Discussion

There are always limits on rule bases; if the goal is to find a rule base so that Deep Blue can beat Kasparov, then the rule base can be complicated and the constraints on it (e.g., 40 moves in two hours so it can't take forever to evaluate positions) may be modest; if the goal is to convey rules to Private Schmucatelli in a few days of training so that he can beat Private so-and-so of the red team, the limits are more severe; that is, it may be possible to only convey some basic principles and a grasp of the big picture.

4.1 Side Benefits

Among the advantages of optimization procedures which run lots of rule bases is their debugging value, since model-based methods, especially when augmented by partial randomization, venture into regions of the state space that programmers may not have fully thought through and find things that human analysts don't. Analysts should save the random number seeds that allow them to reproduce runs, so that some rule bases can be run in debug mode. Rule bases which are outliers in the regression are good candidates for use in debugging.

4.2 Future work

The modeling procedure we use does not depend on the way in which the data were obtained. For example, one could fit models using data obtained from a genetic algorithm or other optimization procedure. These models could then contribute to the evolution of the genetic material (for instance, by suggesting rule bases for immigration).

This simulation is simplified in many ways, and most of these simplications will be relaxed in future research. All status variables and all action variables are binary; our methodology could be easily adapted to handle polychotomous variables, but the size of the relevant spaces would get potentially much larger. Odds ratios can still be formed for $I \times J$ tables (Agresti 1990), but there are problems with uniqueness and we have no experience using such ratios in regressions. Agents never run out of ammunition, so that the optimal rule base can recommend shooting always, rather than having to determine potentially complicated strategies for conserving bullets for situations in which they have larger hit probabilities. Agents have no sensors and never communicate with each other, so agents do not have to take into account information about areas they cannot see nor do they have to work in concert with other agents. Agents are either alive or dead: in other words, wounded soldiers do not suffer from partially degraded performance; agent health could be another dimension of status. There is no terrain to complicate movement or line of sight, and all blue agents have the same rule base regardless of their roles, limiting the size of the problem. It is also of interest to use the same simulation code to solve problems involving how to equip forces, and so on; we do not discuss these problems here. Finally, military tactics of each side co-evolve in game-theoretic fashion, with each side reacting to the other.

Steve: could use some references here to more realistic codes The point here, however, is to consider a simple situation where it is fairly clear what's happening and where the runtimes are short enough to allow reasonable exploration of the space.

Note that the concept of a "best" rule base may depend in part on the underlying variability. Consider the case where the blue team wants to maximize the probability of killing all of the red agents. When the blue team can win most of the time, it is often to its advantage to decrease the spread of outcomes. On the other hand, if the blue team usually loses, it may want to increase the spread to give it the best chance of winning. In the case at hand, it turns out that good rule bases lead to over half of the blue agents reaching the goal, so that maximizing the average is a reasonable thing to do.

Bayesian methods for classification and regression trees (Chipman, George, and McCulloch 1998; Denison, Mallick, and Smith 1998) could also be modified for optimization. An issue to be overcome is that problem is "backwards" from the usual CART approach; there, have a fixed data set and want to find a tree that gives best agreement; here, start with trees and do so-called data farming to grow more data; thus, overfitting isn't quite the problem that it usually is in CART applications in that can always generate additional data; there could be something in the methods used, though, to get better trees; such an approach would still have to deal with the fact that fitness isn't a smooth measure of tree-space variables;

when co-evolving, there's an infinite regress component to deal with: once blue optimizes relative to red, then red will adjust to blue's new tactics, which can be done using the same approach as here; at that point, blue must adjust to red's new tactics, and the process evolves;

close on a positive note, emphasizing the advantages relative to working in bit string space and being able to learn simple rule

Appendix: JIVES Town Simulation

Below we discuss the simulation framework we studied as part of this research. For more detail, see (*** TSA-5 report).

The scenario we studied is depicted in Figure (whatever). The blue agents must move from their starting position at the right side of the figure to the goal at the left side. The red agents are in a defensive mode, stationed along a side street to prevent blue progress. They have a fixed rule base, remaining in their starting locations, always in a defensive posture, always looking, and shooting whenever they sense a blue agent. The blue agents' rule base can be changed: they may move, look, shoot or not, they may change their posture, and if they are both shooting and changing posture, they can invert the order in which these actions are carried out. Defensive posture decreases one's speed by half and also decreases the opponents' probability of a successful shot by half.

In the simple case considered here, all blue agents consult the same rule base. The rule base is defined by the five variables at the current time step, and the rule base determines the action(s) taken by the agent solely as a function of current status. Actions taken do not depend on past events.

Whenever two agents are within a threshold distance of each other they have a probability of seeing each other which depends on their distance, should they choose to look. (IS THIS RIGHT? NO MEMORY?) Another stochastic component of the code is the kill probability p_k: if an agent shoots at an enemy agent (which he has sensed), the probability of a kill behaves exponentially as a function of distance. (Steve: any references to support that this model is a good one?)

Some of the rules depend on others: for example, one must look during one time step in order to sense on the next, and one must sense an enemy agent in order to shoot. The "change firing order" action is meaningful only if the agent is both shooting and changing posture. Agents must move in order to reach their goals. These dependencies imply that some rule bases can be interpreted as impossible or as the same as some other ostensibly different rule bases.

This code contains some simplifications:

(a) looking neither slows agents down nor exposes them to enemy fire, so it is always best to look; Also, a "look" action includes an attempt to sense in every direction, so that agents don't need to explicitly turn their heads;

(b) shooting doesn't reveal one's location and ammunition is infinite, so it is always best to shoot; and

(c) agents move in a straight line along a predetermined path at a constant rate, as opposed to deciding on direction/speed.

When optimizing rule bases, it is not always appropriate to act as if one has no knowledge about good actions, particularly if the action space is larger than in this example. Some of the simplifications discussed above could lead to simplifications in the rule bases under consideration.

Natural enhancements to this scenario include actions with more than two levels (for instance, move at rapid, moderate, slow, or zero speed; the actual speed that each of these would mean could depend on the terrain). Some scenarios include obstacles to line-of-sight. Soldiers could be injured and thus impaired, and their rule bases could change as a result. Finite ammunition and communication could also make things more complex. In this code, agents are incapable of determining whether other agents are alive or dead, so that all enemy agents of either condition are eligible to be shot at. This avoids some errors of recklessness; it might be more realistic for agents to shoot preferentially at enemies who are obviously alive, if there are any.

References

Agresti, A. (1990), *Categorical Data Analysis*, New York: John Wiley.

Bishop, Y. M., Fienberg, S. E., and Holland, P. W. (1975), *Discrete Multivariate Analysis.* Cambridge: MIT Press.

Chipman, H. A., George, E. I., and McCulloch, R. E. (1998), "Bayesian CART Model Search" *Journal of the American Statistical Association* **93** 935-947.

Deadman, P. J. (1999), "Modelling Individual Behaviour and Group Performance in an Intelligent Agent-Based Simulation of the Tragedy of the Commons," *Journal of Environmental Management* **56** 159-172.

Denison, D. G., Mallick, B. K., and Smith, A. F. M. (1998), "A Bayesian CART Algorithm," *Biometrika* **85** 363-377.

Fogerty, T. C. and Bull, L. (1995), "Optimising Individual Control Rules and Multiple Communicating Rule-Based Control Systems with Parallel Distributed Genetic Algorithms," *IEE Proceedings - Control Theory and Applications* **142** 211-215.

Jaffe, K., Issa, S., Daniels, E., and Haile, D. (1997), "Dynamics of the Emergence of Genetic Resistance to Biocides among Asexual and Sexual Organisms," *Journal of Theoretical Biology* **188** 289-299.

Murthy, S. K. (1998), "Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey," *Data Mining and Knowledge Discovery* **2** 345-389.

Nagel, K., Rickert, M., and Barrett, C. L. (1997), "Large Scale Traffic Simulations," *Lecture Notes in Computer Science* **1215** 380-402.

Samuels, M. L. (1993), "Simpson's Paradox and Related Phenomena" *Journal of the American Statistical Association* **88** 81-88.